Санкт-Петербургский государственный университет

Губайдуллин Булат Альбертович

Выпускная квалификационная работа

Диагностика заболеваний по электрокардиограмме с использованием методов машинного обучения

Уровень образования: *бакалавриат*

Направление 01.03.02 *“Прикладная математика и информатика”*

Основная образовательная программа СВ.5005.2016 “*Прикладная математика, фундаментальная информатика и программирование”*

Профиль *“Процессы управления и высокопроизводительные вычислительные системы”*

Научный руководитель:

доцент кафедры ТСУЭФА,

к.ф.-м.н.,

Головкина А. Г.

Рецензент:

профессор кафедры ТУ,

д.ф.-м.н.,

Котина Е.Д.

Санкт-Петербург

2020

Оглавление

Введение

Сердце - главный орган в организме любого живого существа, отвечающий за все его жизненно важные функции. Работая как насос, он осуществляет перекачку крови и распространению ее по всему организму для насыщения кислородом и поддержания в теле теплового баланса. Любые отклонения в работе сердечной мышцы постепенно сказываются на работе всего организма в целом. По официальной статистике Всемирной Организации Здравоохранения, сердечно-сосудистые заболевания (ССЗ) являются главной причиной смерти во всём мире [19], поэтому задача правильной диагностики и лечения таких заболеваний является актуальной.

Электрокардиографией называется метод исследования и регистрации электрической активности сердца для диагностики различных патологий [11]. История развития электрокардиографии уходит в далёкий 1842 год, когда немецким физиологом Дюбоис-Реймондом были обнаружены электрические сигналы, испускаемые при сокращениях сердца лягушки. Это открытие положило начало исследованиям электрической активности сердечной мышцы. Позднее, в 1872 году английским учёным Огустом Валлером была записана электрическая активность этого органа с использованием капиллярного электрометра люксембургского физика Габриэля Липпмана, а в 1887 году на международном конгрессе физиологов в Лондоне голландский физиолог Вильям Эйнтховен провёл демонстрацию кривой потенциала действия сердца человека, записанной на изобретённом им струнном гальванометре. Далее последовала череда знаменательных открытий в области электрокардиографии, причём многие результаты которых используются и в наши дни.

Однако, несмотря на то что современная медицина позволяет с высокой точностью диагностировать практически любые отклонения в правильной

работе сердца, анализ электрокардиограммы и постановка диагноза осуществляются медицинским работником, и этот факт, даже невзирая на профессионализм и многолетний опыт работы отдельных специалистов, не позволяет полностью исключить фактор человеческой ошибки. Комбинация различных видов заболеваний, внешние шумы в сигнале, специфичность заболевания - эти и множество других факторов могут стать существенным препятствием в постановке правильного диагноза, что может стоить жизни пациенту.

В цифровую эпоху множество сфер человеческой деятельности оказываются автоматизированы - люди пытаются упростить, минимизировать ручную работу и оптимизировать рабочие процессы, предоставив решение этих задач компьютерам. Компьютерные алгоритмы позволяют обрабатывать информацию гораздо быстрее, чем это может сделать человек. Особенно это становится заметно, когда дело касается больших объемов данных. Сердце человека как раз является “генератором” таких данных - из его последовательных сокращений можно получить достаточно объемный набор информации, который может быть обработан машиной, при этом обработка займёт гораздо меньше времени, чем анализ человеком. Таким образом возникает вопрос - возможно ли реализовать программное средство, способное производить диагностику сердечной мышцы и диагностировать патологии, помогая медработнику принять решение и одновременно минимизируя его непосредственное участие?

Реализация данного программного комплекса состоит из двух задач. Первой задачей является препроцессинг - предварительная обработка данных для представления их в удобном виде. Его принципы описаны в первой главе. Не всегда данные имеют общий формат, а также зачастую из необработанных данных невозможно выделить общие для каждого элемента характеристики.

Препроцессинг призван решить эту проблему и преобразовать данные таким образом, чтобы их они стали пригодными для использования в дальнейшем этапе исследования.

Во второй главе описывается разработка алгоритма, способного осуществлять диагностику. В данном случае под диагностикой фактически понимается решение задачи классификации. Существует ряд алгоритмов, получающих на вход некий набор данных и возвращающих информацию о том, к каком из определенных классов эти данные соответствуют [20]. В качестве примера можно привести искусственные нейронные сети - математические модели, основанные на принципах организации биологических нейронных сетей. В наши дни подобные алгоритмы используются для обширного класса задач: управления, распознавания образов, прогнозирования и многих других [20]. В этом исследовании будет описан процесс создания и использования одной из таких моделей для анализа полученных на этапе препроцессинга данных электрокардиограмм и установления факта принадлежности ее к одному из заданных классов. В работе, в качестве примера, рассматривается три класса. Два из них соответствуют наиболее распространенным сердечно-сосудистым заболеваниям (аритмия, инфаркт), по которым накоплен достаточно большой объем электрокардиографических исследований, а третий класс соответствует электрокардиограммам без патологических изменений.

Постановка задачи

Целью данной работы является реализация программного комплекса для диагностирования инфаркта и аритмии по электрокардиограмме (ЭКГ) с использованием методов машинного обучения. Для достижения поставленной цели необходимо решить следующие задачи:

- Найти набор данных, содержащий достаточное количество примеров ЭКГ с состояниями инфаркта, аритмии, а также ЭКГ здоровых людей;
- Осуществить обработку данных, заключающуюся в удалении шумов и смещений с использованием известных алгоритмов фильтрации сигнала;
- Определить ключевые точки на ЭКГ (положения зубцов);
- Преобразовать полученные значения в набор характеристик, сохранить данные в формате, удобном для дальнейшего использования;
- Разработать модель искусственной нейронной сети, получающую на вход вышеописанный набор данных;
- Обучить модель на этих данных и оценить полученные результаты.

Обзор литературы

При реализации каждой части системы (препроцессинг и модель нейронной сети) был проведен анализ существующей литературы, среди которой особо стоит отметить [4-5, 7, 10, 15]. В результате были выбраны наиболее оптимальные методы решения поставленных задач.

Для решения задачи препроцессинга были рассмотрены источники [1-3, 8, 11-12, 16], в которых представлены способы анализа электрокардиограмм для корректного определения медицинских показателей, а также электронные ресурсы баз данных ЭКГ [13, 14], в которых представлены различные оцифрованные сигналы. В данном исследовании было принято решение использовать базу данных [13], так как ее содержимое лучше отвечало условиям поставленной задачи.

Ресурсы [6, 9, 17-18] описывают некоторые методы решения второй задачи - реализации алгоритма классификации. Несмотря на то, что идеи решения проблем классификации описаны во многих исследованиях, именно эти источники представляют особый интерес из-за специфики задачи.

Глава 1. Обзор данных и их препроцессинг

В данной главе рассмотрена выбранная для исследования база данных, методы обработки зашумленного сигнала, а также алгоритм извлечения из кардиосигнала значимых для диагностики характеристик.

§1.1 PTB Diagnostic ECG Database

Для реализации поставленной задачи была выбрана база [13], сформированная на основе данных, собранных исследователями из Национального Метрологического института Германии. Она состоит из более чем 500 оцифрованных электрокардиограмм от более чем 250 человек, среди которых есть как люди с реальными заболеваниями, так и здоровые волонтеры без патологических изменений на электрокардиограмме. Данный набор содержит информацию по различным типам патологий сердца, однако в связи с некоторыми ограничениями в возможности корректного распознавания отдельных заболеваний, которые будут рассмотрены далее, а также из-за недостаточного количества электрокардиограмм, отражающих отдельные виды заболеваний, было принято решение ограничиться лишь двумя видами - аритмией и инфарктом миокарда.

На Рис.1 изображен пример оцифрованного сигнала ЭКГ (отведение I) из рассматриваемого датасета, записанного с частотой 1000 Гц. В ЭКГ существует множество отведений (основные отведения I, II, III, грудные V1-V6 и несколько дополнительных), с помощью которых по-разному можно оценивать электрический потенциал сердечной мышцы, но в данном исследовании было рассмотрено отведение I.

Как можно заметить, сигнал достаточно сильно зашумлен, что может значительно затруднить диагностику. Кроме того, если рассмотреть более

длинный интервал изменений данного сигнала (Рис.2), будет явно видно смещение (еще иначе его называют трендом) сигнала во времени. В следующих параграфах мы рассмотрим метод очистки сигнала от тренда и его фильтрацию.

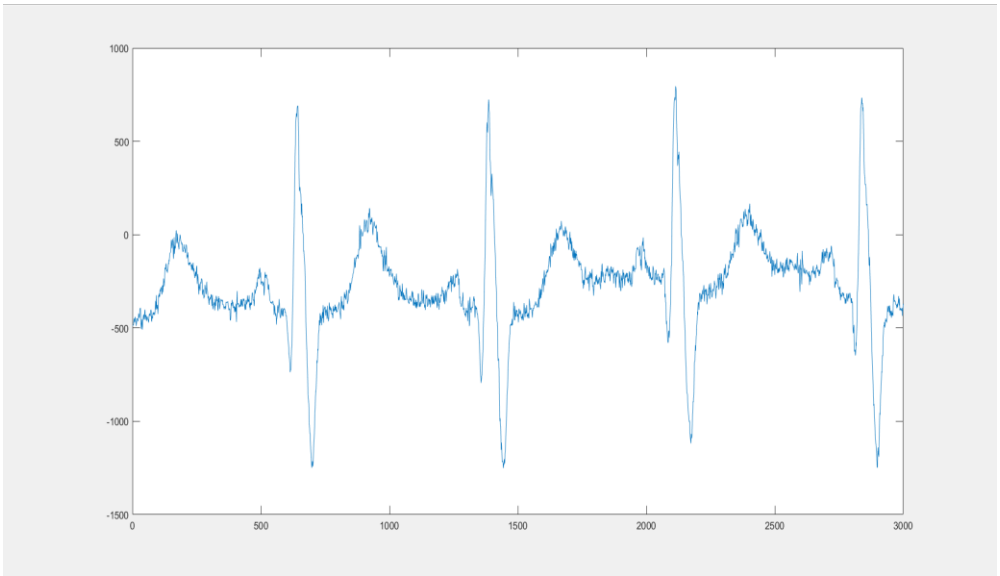

Рис. 1. Пример сигнала из датасета PTB Diagnostic ECG Database (длительность - 3 сек.)

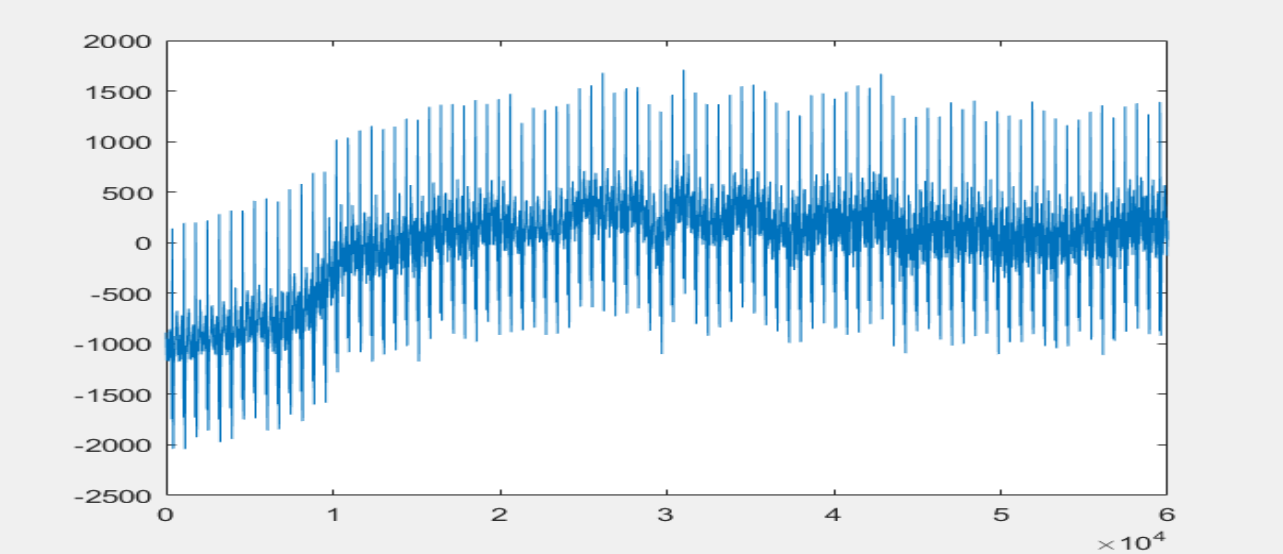

Рис. 2. Пример сигнала, изображенного на Рис.1 (длительность - 60 сек.)

§1.2 Детрендинг сигнала

Детрендингом называют процесс удаления из сигнала его смещений во времени. Эта процедура позволяет получить из исходного сигнала новый, обладающий двумя полезными свойствами:

- В нем будет практически отсутствовать смещение;
- Сигнал станет “нормированным” (будет выровнен к нулю по оси ординат).

Обозначим исходный сигнал функцией *y(t)*. Пусть в сигнале присутствует линейное смещение $d(t) = a*t + b$. Вычитая из исходного сигнала *y(t)* смещение *d(t)*, мы получим новую функцию $z(t) = y(t) - d(t)$, которая представляет собой очищенный от тренда сигнал. На Рис.3 изображен пример удаления из сигнала линейного тренда. Сплошной синей линией обозначен исходный сигнал *y(t)*, пунктирной линией - смещение *d(t)*, сплошной красной - полученный несмещенный сигнал *z(t)*.

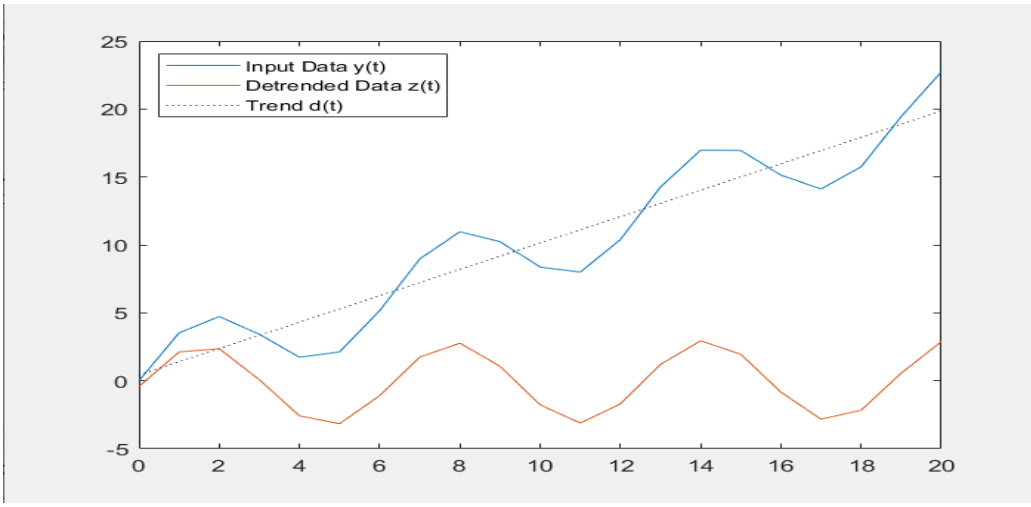

Рис. 3. Удаление линейного тренда из смещенного во времени сигнала

Однако не всегда смещение является линейным. На Рис.4 изображен сигнал *x(t)* с квадратичным смещением *d(t)*.

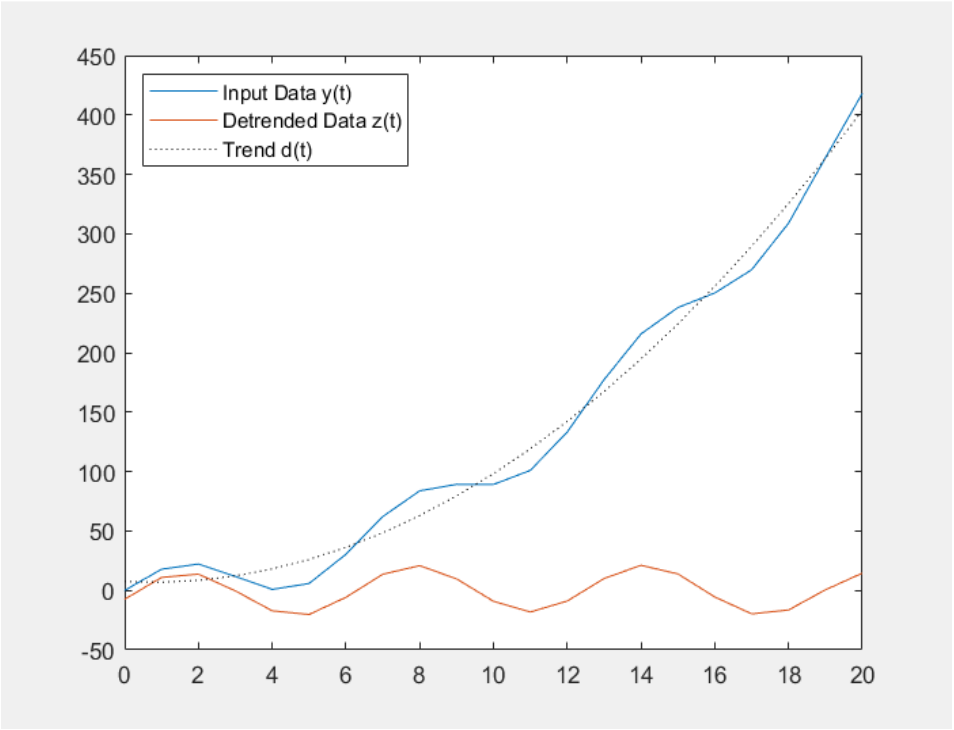

Рис. 4. Удаление квадратичного тренда из смещенного во времени сигнала

Однако из Рис.2 видно, что в исходном сигнале смещение является более сложным, чем линейное или квадратичное, и оно представляет собой полином некоторого высокого порядка. Для того, чтобы удалить смещение из сигнала, необходимо найти коэффициенты аппроксимирующего данный сигнал полинома.

Аппроксимирующий полином степени n можно представить в следующем виде: $p(x) = p_1x^n + p_2x^{n-1} + ... + p_nx + p_{n+1}$, где p_m, m ∈ [1, n+1] - искомые коэффициенты. Для нахождения коэффициентов строится система, изображенная на Рис.5. В данном случае мы полагаем:

- x_i , i ∈ [1, m] - набор точек, по которым вычисляется приближение;
- y_i , i ∈ [1, m] - значение исходной функции в этих точках.

$$\begin{pmatrix} x_1^n & x_1^{n-1} & \cdots & 1 \\ x_2^n & x_2^{n-1} & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ x_m^n & x_m^{n-1} & \cdots & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n+1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Рис. 5. СЛАУ для нахождения коэффициентов аппроксимирующего полинома

Матрица X в левой части системы называется матрицей Вандермонда. Решением системы является вектор $p = (p_1, p_2, \ldots, p_{n+1})^T$, вычисляемый по формуле $p = y*X^{-1}$.

Таким образом, вычитая из исходного сигнала $y(t)$ полином $p(t)$, мы получим очищенный от смещения сигнал. На Рис.6 можно видеть применение данного метода детрендинга к сигналу, изображенному на Рис.2. Программный код данного алгоритма приведен в Приложении 1.

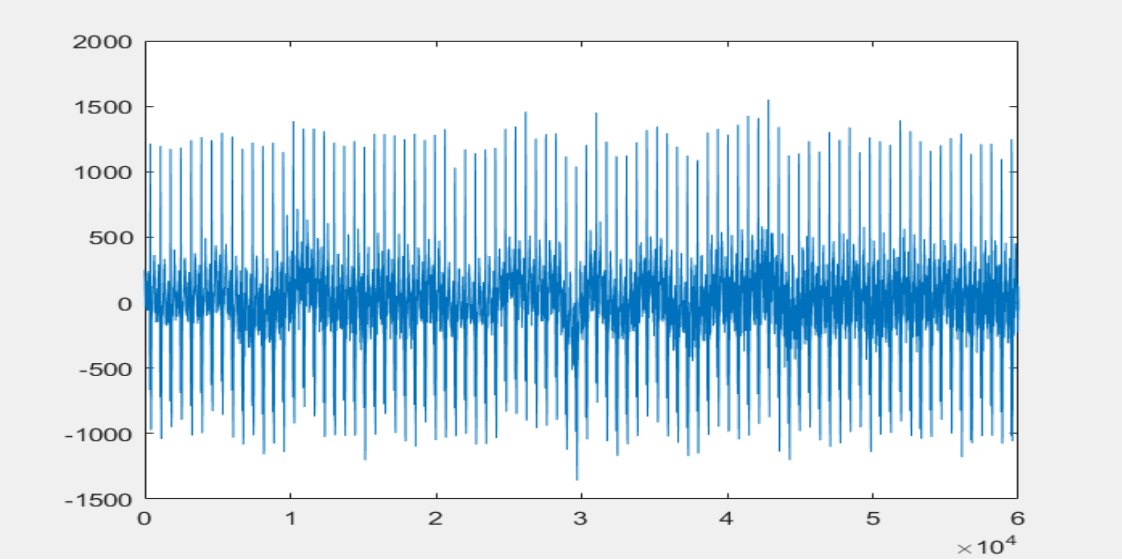

Рис. 6. Очищенный от тренда сигнал, изображенный на Рис.2 ($p(x)$ степени 10)

§1.3 Фильтрация методом Савицкого-Голея

Цель фильтрации - удалить из сигнала шумы, не связанные со смещением во времени. В общем случае, можно сказать, что сигнал задан следующим образом: $x(t) = S(t) + n(t)$, где:

- $x(t)$ - зашумленный сигнал;
- $S(t)$ - полезная составляющая сигнала;
- $n(t)$ - шум.

Тогда возникает задача построения оптимального фильтра, для того чтобы разделить сигнал $x(t)$ на составляющие $S(t)$ и $n(t)$.

Существуют различные методы, предназначенные для решения указанной задачи. В данном исследовании рассматривается фильтр Савицкого-Голея - цифровой фильтр (фильтр, работающий с дискретным во времени (discrete-time) сигналом), который может быть применен к набору точек оцифрованного сигнала, для того чтобы сгладить его, не искажая его вид. Для этого применяется следующая идея: приближение к сигналу строится на основе суммы значений исходного сигнала, умноженных на некоторые коэффициенты (этот алгоритм называется сверткой). Построенная последовательность приближений будет являться сглаженным сигналом.

Сверткой называется математическая операция над двумя функциями, производящая третью функцию, выражающую, как вид одной функции будет модифицирован другой. В общем виде свертка функций f и g записывается как $f * g$, и имеет следующий вид (для непрерывных систем):

$$(f * g)(t) \triangleq \int_{-\infty}^{\infty} f(\tau) g(t - \tau)\, d\tau.$$

Для дискретных систем:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Рассмотрим следующую функцию свертки:

$$Y_j = \sum_{i=\frac{1-m}{2}}^{\frac{m-1}{2}} C_i \, y_{j+i}, \qquad \frac{m-1}{2} \le j \le n - \frac{m-1}{2}$$

Здесь:

- y_s, s ∈ [0, n] - значения исходного сигнала;
- Y_j - значения полученного приближения;
- C_i - коэффициенты свертки;
- m - размер окна свертки (количество рассматриваемых точек).

Коэффициенты свертки выбираются таким образом, чтобы полином степени k, построенный точкам y_s, был оптимальным в смысле наименьших квадратов, то есть $\Sigma_{n=1..s}(y_s-y_s')^2 \to min$ (сумма расстояний от точек y_s до точек аппроксимирующего полинома y_s' должна быть минимальной).

Для оптимизации решения задачи о наименьших квадратах применяется алгоритм Левенберга-Марквардта [21]. Пусть имеется следующая постановки задачи о наименьших квадратах:

$$F(\vec{x}) = \|\vec{f}(\vec{x})\|^2 = \sum_{i=1}^{m} f_i^2(\vec{x}) = \sum_{i=1}^{m} (\varphi_i(\vec{x}) - \mathcal{F}_i)^2 \to \min$$

Данная задача примечательна специальным видом градиента и гессиана:

$$\nabla F(\vec{x}) = 2J^T(\vec{x})\vec{f}(\vec{x}),$$

$$H(\vec{x}) = 2J^T(\vec{x})J(\vec{x}) + 2Q(\vec{x}), \qquad Q(\vec{x}) = \sum_{i=1}^{m} f_i(\vec{x})H_i(\vec{x}),$$

$$H(\vec{x}) = 2J^T(\vec{x})J(\vec{x}) + 2Q(\vec{x}), \qquad Q(\vec{x}) = \sum_{i=1}^{m} f_i(\vec{x})H_i(\vec{x}),$$

где:

- $J(x)$ - матрица Якоби вектор-функции $f(x)$;
- $H_i(x)$ - матрица Гессе для ее компоненты $f_i(x)$.

Тогда, по методу Гаусса-Ньютона, очередное направление p системы вычисляется следующим образом:

$$J^T(\vec{x})J(\vec{x})\vec{p} = -J^T(\vec{x})\vec{f}(\vec{x}).$$

На Рис. 7 изображен результат применения фильтра Савицкого-Голея с параметрами к сигналу, изображенному на Рис.1 (слева - исходный сигнал, справа - полученный). Параметры фильтрации сигнала ($k = 7$ и $m = 61$) были выбраны экспериментально. Как видно, данный метод фильтрации очищает сигнал достаточно хорошо.

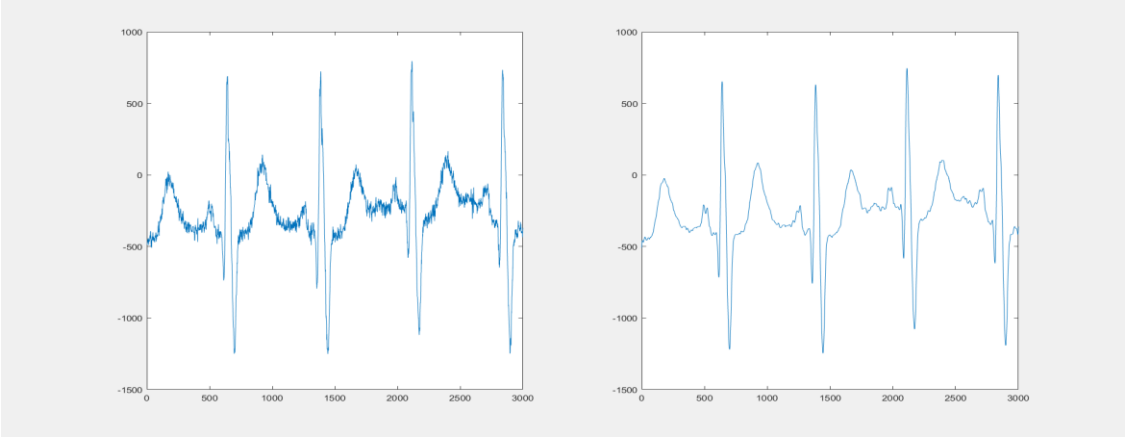

Рис. 7. Применение алгоритма Савицкого-Голея к фильтрации сигнала

Таким образом, применение двух методов обработки цифрового сигнала позволяет нам получить очищенный сигнал, поддающийся дальнейшей обработке.

§1.4 Характеристики ЭКГ

Любая (за редкими исключениями) ЭКГ отражает важные медицинские характеристики - сокращения камер сердечной мыщцы, которые можно наблюдать по характерным зубцам. Их описание можно найти в [1-3]. Существует пять основных зубцов - *P*, *Q*, *R*, *S* и *T*, а также еще один зубец *U*, который на практике встречается редко, поэтому обычно не рассматривается. На Рис.8 изображен кардиоцикл. Кардиоцикл показывает последовательность процессов, происходящих за одно сокращение сердца и его последующее расслабление. Зубец *P* означает сокращение предсердий (наполнение сердца кровью), комплекс *QRS* - сокращение желудочков (выброс крови) и зубец *T* (расслабление желудочков). Поведение этих зубцов определяет состояние сердца и эффективность его работы.

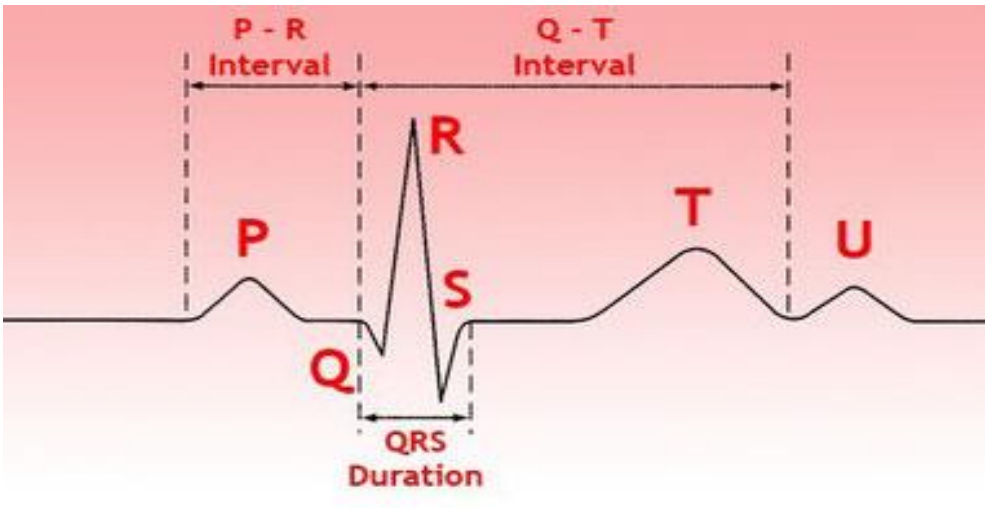

Рис. 8. Стандартный вид кардиоцикла в ЭКГ

Совокупность положений зубцов по каждому кардиоциклу однозначно определяет кардиограмму, поэтому следующей задачей является определение координат каждого из пяти основных зубцов.

§1.5 Алгоритм определения положений зубцов

Стандартный кардиоцикл рассматривается от начала зубца *P* до конца зубца *T* (или *U*, если он имеется). Для удобства анализа мы отойдем от этого определения и будем называть кардиоциклом интервал от одного зубца *R* до следующего (так называемый *RR*-интервал). На Рис.9 изображен пример одного кардиоцикла.

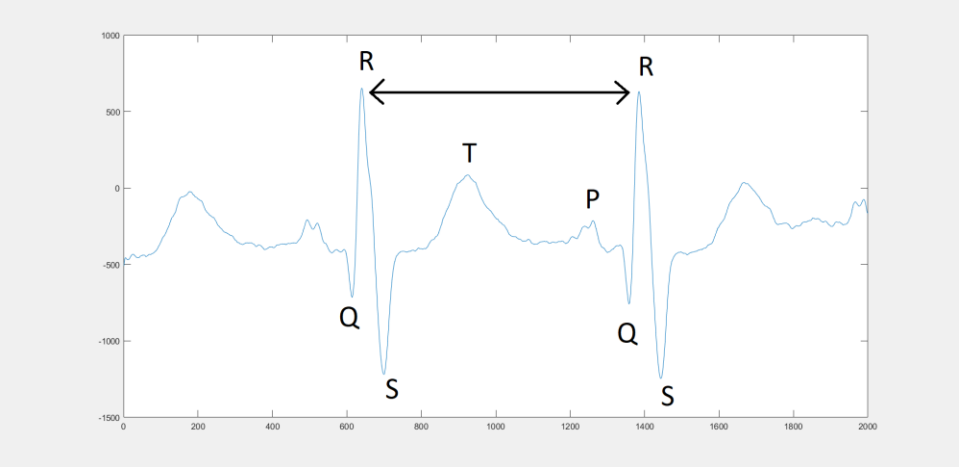

Рис. 9. RR-интервал на кардиограмме

Рассмотрим процесс определения положений зубцов в сигнале. Зубцы *R* (непосредственные моменты удара сердца) самые высокие, поэтому определение их положения не составляет труда - необходимо просто найти все максимумы на кардиограмме, которые находятся выше некоторого значения (для каждой кардиограммы может быть своё):

R_locations = peaks(cardioSignal(x), где x > x_critical)

Найдя координаты всех зубцов *R*, разделим сигнал на кардиоциклы - множество кусочков кардиограммы от одного зубца *R* до другого. Все кардиоциклы похожи друг на друга, поэтому достаточно разработать алгоритм определения оставшихся зубцов для одного кардиоцикла и применить его ко всем остальным.

Можно заметить интересную особенность оставшихся зубцов - их положение является локальным максимумом и минимумом в некоторой ограниченной области кардиоцикла. Предлагается следующая идея: разделим кардиоцикл на две равные части. Обозначим:

- *leftCardioSignal(x) = cardioSignal(x), где x < length(cardioSignal)*
- *rightCardioSignal(x) = cardioSignal(x), где x > length(cardioSignal)*

Тогда:

- *S = min(leftCardioSignal);*
- *T = max(leftCardioSignal);*
- *P = max(rightCardioSignal);*
- *Q = min(leftCardioSignal).*

Помимо положений зубцов, существуют также другие значимые характеристики кардиограммы - размеры всех зубцов, а также длины интервалов *PQ*, *QT* и некоторых других. Для определения этих параметров необходимо найти границы каждого из зубцов, причем зубцы *P* и *T* имеют две границы (левую и правую), а зубцы *Q* и *S* - лишь по одной (*Q* - только левую, *S* - только правую).

Для определения положений границ зубцов *P* и *T* применяется следующая техника - слева и справа от положений этих зубцов анализируется знак производной. Для нахождения левой границы (на примере зубца *P*) алгоритм следующий:

1) Рассмотрим некоторую область слева от зубца *P*, обозначим ее *y(t)*;
2) Выбираем точку такую, в которой *|y' (t)| -> min, y' (t) > 0.*

Для правой границы алгоритм аналогичный, только выбираем область справа от P, и во втором шаге ищем точку такую, что *y' (t) < 0.*

Для определения границ зубцов *Q* и *S* применяется аналогичная идея. Рассмотрим на примере зубца *S*. Обозначая через *y(t)* область справа от *S*, находим точку, для которой $|y'(t)| \to min, y'(t) > 0$. Аналогично для *Q*, ищем слева от *Q* точку такую, что $|y'(t)| \to min, y'(t) < 0$.

Таким образом, применяя эти принципы к каждому кардиоциклу, можно найти все ключевые точки в кардиограмме. Однако, несмотря на тот факт, что этот метод срабатывает в большинстве случаев, есть ряд особых случаев, представленных на рисунках 10-12, которые необходимо отметить отдельно.

Рис. 10. Пример аномалии - трепетание и фибрилляция желудочков

Рис. 11. Пример аномалии - асистолия

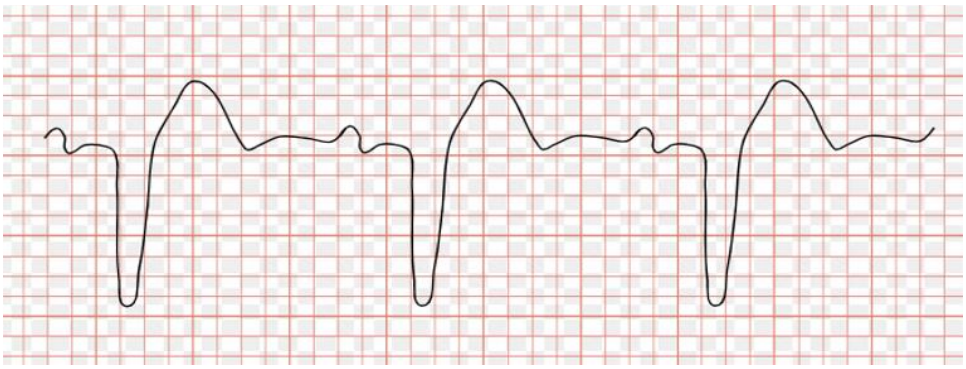

Рис. 12. Пример аномалии - блокада ножек пучка Гиса

Главной проблемой при анализе таких аномалий является то, что для них комплекс *QRS* либо с трудом поддается определению (Рис.12), либо отсутствует совсем (Рис.10-11), таким образом из них невозможно выделить информацию, поддающуюся анализу.

Рассмотрим еще один примечательный тип аномалий - выпадение зубца *T*. Как видно из Рис.8, зубец *T* в норме направлен вверх (другими словами, он является положительным). Однако при некоторых болезнях он может выпадать вниз (отрицательный зубец *T*). Пример этого явления приведен на Рис.13.

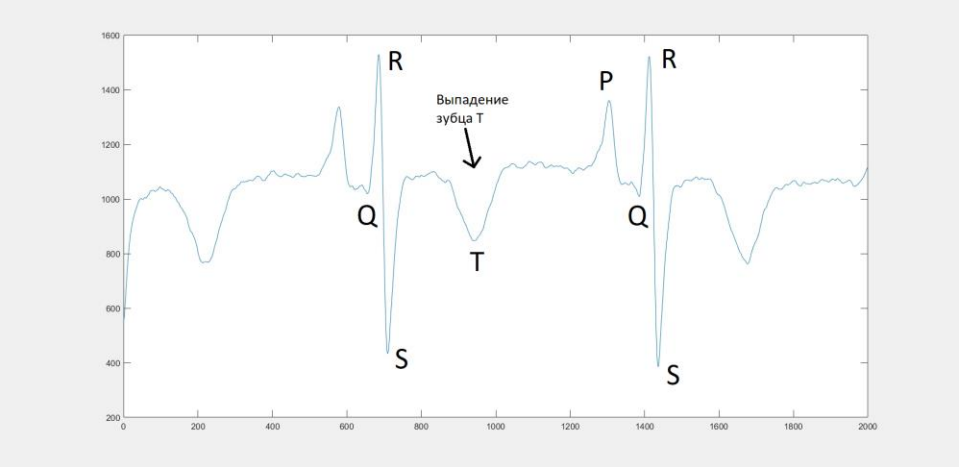

Рис. 13. Пример отрицательного зубца Т.

Данная аномалия может возникать из-за некоторых типов нарушения проводимости сердца. На данном рисунке изображен кардиоцикл кардиограммы с инфарктом миокарда, однако это не означает, что аномалия зубца *T* имеет место при такой болезни сердца. Тем не менее, так как такое поведение зубца *T* нарушает вышеописанную идею о том, что зубец *T* определяется как максимум в левой части кардиоцикла, это стоит принимать во внимание.

Для определения положения отрицательного зубца *T* необходимо проанализировать знак производной в районе его местоположения. Если производная меняет свой знак с “-” на “+”, тогда мы имеем дело с аномальным зубцом Т, иначе - с нормальным.

§1.6 Полученные результаты и преобразование данных

Результатом применения рассмотренных в этой главе методов очистки сигнала от шума, а также анализа положений зубцов в кардиоцикле является разработанный на языке программирования MATLAB алгоритм, позволяющий автоматически определять все значимые характеристики. На Рис.14-15 представлен результат работы этого алгоритма.

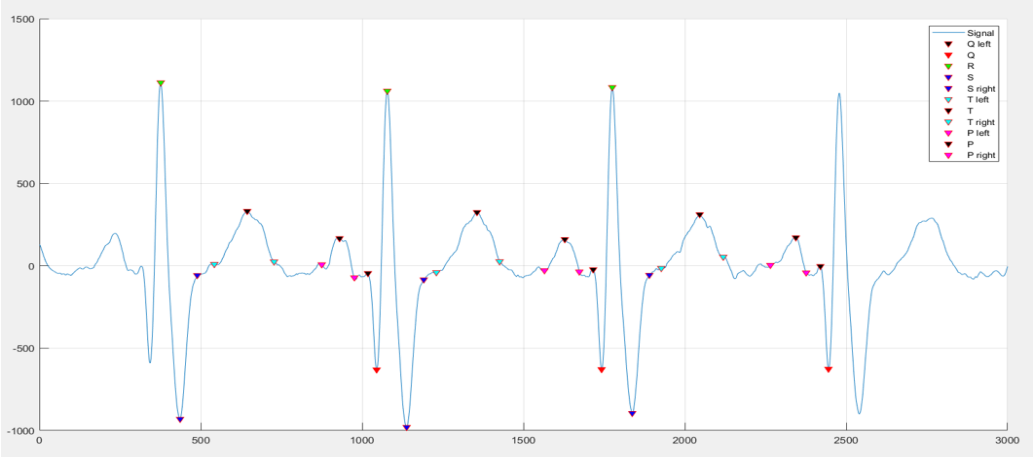

Рис. 14. Результат определения зубцов на ЭКГ с нормальным зубцом T

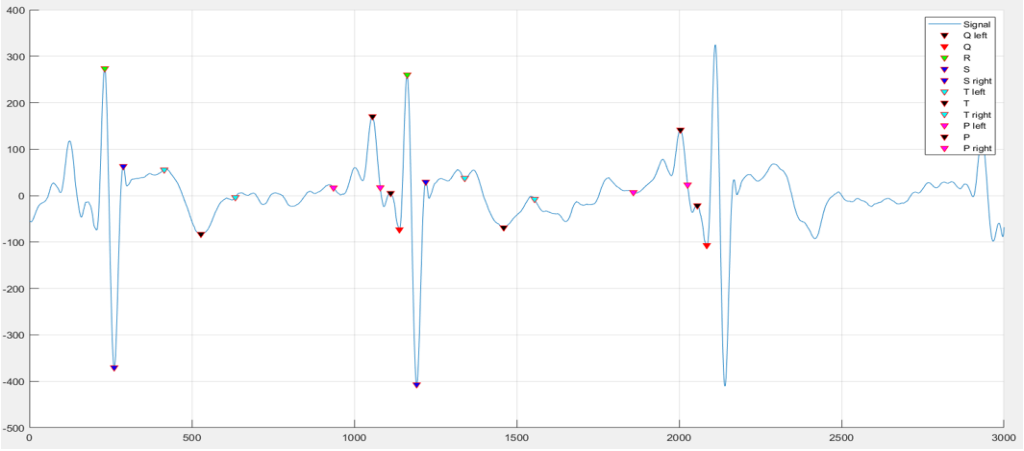

Рис. 15. Результат определения зубцов на ЭКГ с аномальным зубцом T

Последним этапом препроцессинга является преобразование полученных данных в некоторые важные параметры, определяющие эффективность работы сердца. Согласно медицинской литературе по кардиологии [1-3], такими параметрами являются:

- Длины интервалов *RR*, *QT*, *PQ*, комплекса *QRS*;
- Амплитуды зубцов *S*, *P* и *Q*;
- Размер зубца *Q*;
- Направление зубца *T* (положительный/отрицательный).

Обозначим левую границу зубца *Q* как *Q_left*, правую границу зубца *S* как *S_right*, а также правую и левую границы зубцов *P* и *T* как *P_left*, *P_right*, *T_left* и *T_right* соответственно. Тогда:

- Длина интервала *RR (RR_len)* = расстояние между зубцами *R*;
- Длина *QT (QT_len) = T_right – Q_left*;
- Длина *PQ* (PQ_len) = *Q – P_left*;
- Длина *QRS (QRS_len) = S_right – Q_left*;

Амплитуды, направления и размеры зубцов для получения остальных характеристик известны заранее, так как найдены точные координаты зубцов.

Таким образом, собранный набор данных позволяет нам разработать алгоритм, способный на их основе проводить классификацию кардиограмм по типу заболевания.

Глава 2. Многослойный персептрон

Для решения задачи классификации необходимо разработать модель, способную “научиться” отличать здорового человека от больного, а также различать виды болезней. В наши дни очень популярным методом решения подобных задач стали искусственные нейронные сети. В данной главе будет описана структура одной из таких нейросетей на примере многослойного персептрона.

§2.1 Архитектура

Многослойный персептрон — это класс искусственных нейронных сетей прямого распространения, состоящий как минимум из трех слоев: входного, скрытого и выходного. Входной слой получает данные, подлежащие классификации, скрытый слой “обучается” на этих данных, а выходной слой выдает результат принадлежности данных к какому-либо классу. Общий вид многослойного персептрона приведен на Рис.16.

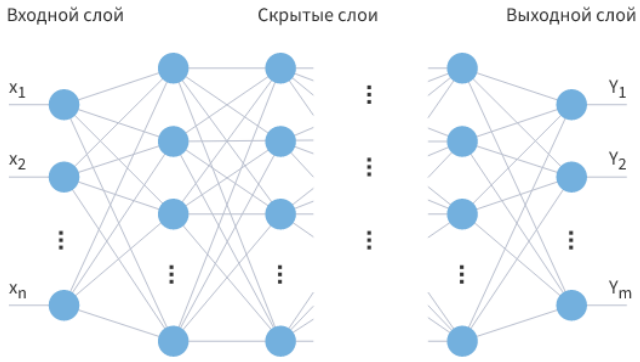

Рис. 16. Многослойный персептрон

Каждый нейрон представляет собой взвешенный сумматор (Рис.17), выход которого определяется через его входы и матрицу весов: $y = f(u)$, где $u = \sum_{i=1}^{n} w_i x_i + w_0 x_0$.

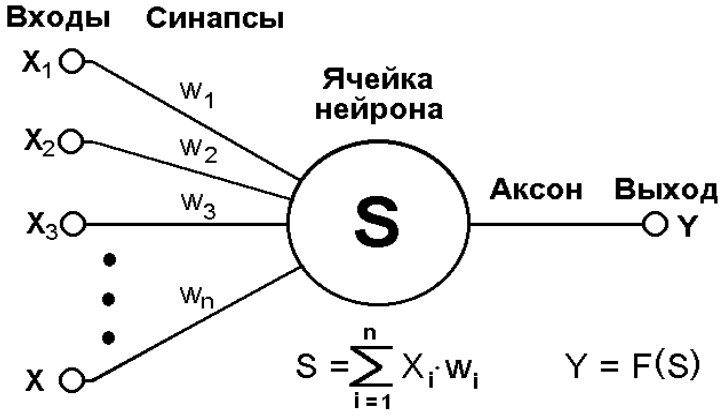

Рис. 17. Стандартная модель нейрона. $w_1 ... w_n$ - веса, F - функция активации

Функция *f(x)* называется функцией активации. Данная функция определяет, будет ли нейрон активирован и каким будет его выходной сигнал. Возвращаемое значение данной функции зависит от набора входных сигналов с предыдущего слоя. Существует множество видов активационных функций. Приведем некоторые из них:

- Тождественная $f(x) = x$
- Логистическая $f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$
- Линейный выпрямитель (ReLU) $f(x) = \begin{cases} 0 & x < 0 \\ x & x \geqslant 0 \end{cases}$
- Линейный выпрямитель с утечкой (Leaky ReLU) $f(x) = \begin{cases} 0,01x & x < 0 \\ x & x \geqslant 0 \end{cases}$

§2.2 Некоторые определения

Эпоха - одна итерация прогона всех данных обучающей выборки через нейронную сеть. От количества эпох зачастую зависит качество обучения сети.

Переобучение - состояние модели, когда она способна адекватно распознавать только тот набор данных, на котором она проходила обучение, но при этом неспособна классифицировать новые данные. Пример такого явления приведен на Рис.18. При переобучении модель “выучивает” выбросы и шум и теряет способность к обобщению, что не позволяет ей адекватно производить классификацию.

Кросс-валидация - перекрестная проверка модели для предотвращения переобучения. Весь набор данных делится на n частей, и далее производится обучение модели, где одна часть используется как тестовое множество, а n-1 остальных - как обучающее. Если модель показывает примерно одинаковые результаты на каждой итерации кросс-валидации, значит модель верно проводит классификацию, в противном случае может иметь место переобучение модели.

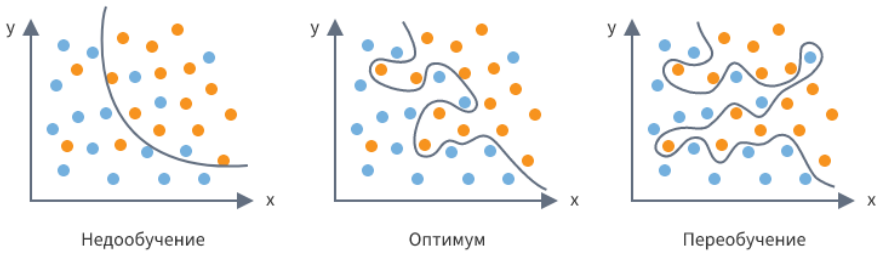

Рис. 18. Пример переобучения нейронной сети

§2.3 Слои сети

1) Входной слой (*Input layer*) - первый слой нейронной сети. Он получает информацию, на основе которой происходит обучение сети. Количество нейронов во входном слое равно количеству параметров в обучающих данных.

2) Полносвязный слой *(Dense*) - это слой, каждый нейрон которого соединяется с каждым нейроном предыдущего и последующего слоев. Этот слой используется как основа для построения многослойного персептрона.

3) Слой исключения (*Dropout*) - данный слой предназначен для того, чтобы предотвратить переобучение сети. Он случайным образом обнуляет весовые коэффициенты некоторых нейронов.

4) Выходной слой (*Output layer*) - этот слой возвращает результат предсказания класса для конкретных входных данных. Количество нейронов в этом слое обычно равняется количеству рассматриваемых в поставленной задаче классов.

Многослойный персептрон состоит из одного входного слоя, за ним следует некоторое количество скрытых слоев (обычно не более трех), затем выходной слой. Между скрытыми слоями иногда находятся слои Dropout, которые не дают модели выучить только обучающий набор и показывать плохие результаты на реальных данных.

§2.4 Метод обратного распространения ошибки

Главной задачей при обучении нейронной сети является минимизация функции ошибки, то есть разности между реальным и полученным ответами. Данный процесс реализуется с помощью метода обратного распространения ошибки, где веса нейронной сетки корректируются с учетом текущего состояния модели, а также величины ошибки.

В основе алгоритма лежит идея использования выходной ошибки сети:

$$E = \frac{1}{2}\sum_{i=1}^{k}(y - y')^2$$

где:

- k - число выходных нейронов сети;
- y - целевое значение;
- y' - фактическое значение.

Настройка весовых коэффициентов производится в соответствии с формулой:

$$\Delta w_{j,i}(n) = -\eta\frac{\partial E_{av}}{\partial w_{ij}}$$

Здесь $\Delta w_{ij}(n) = w_{ij}(n+1) - w_{ij}(n)$, то есть разность между весами текущего и предыдущего шагов обучения.

Таким образом, после каждого прохода сигнала по сети в прямом и обратном направлении, нейронная сеть изменяет свои весовые коэффициенты в сторону уменьшения средней ошибки.

Глава 3. Реализация алгоритма классификации

§3.1 Используемые библиотеки

Разработка комплекса распознавания болезней по ЭКГ производилась на языке программирования Python с использованием следующих библиотек:

- Tensorflow/Keras
- Sklearn
- Pandas
- Matplotlib
- Imblearn
- Numpy
- Seaborn

§3.2 Алгоритм Oversampling

Для корректной работы алгоритма распознавания количество объектов в каждом классе входных данных должно быть приблизительно одинаковым. Однако в полученных входных данных наблюдался дизбаланс в количестве примеров: имелось по 1000 примеров для здоровых людей и людей, больных инфарктом миокарда, и всего лишь 200 для людей с аритмией. Для того чтобы сбалансировать классы, была применена техника [15] Oversampling - случайные объекты класса с малым количеством объектов копировались и добавлялись к общей выборке. Пример применения данной техники можно увидеть на Рис.19.

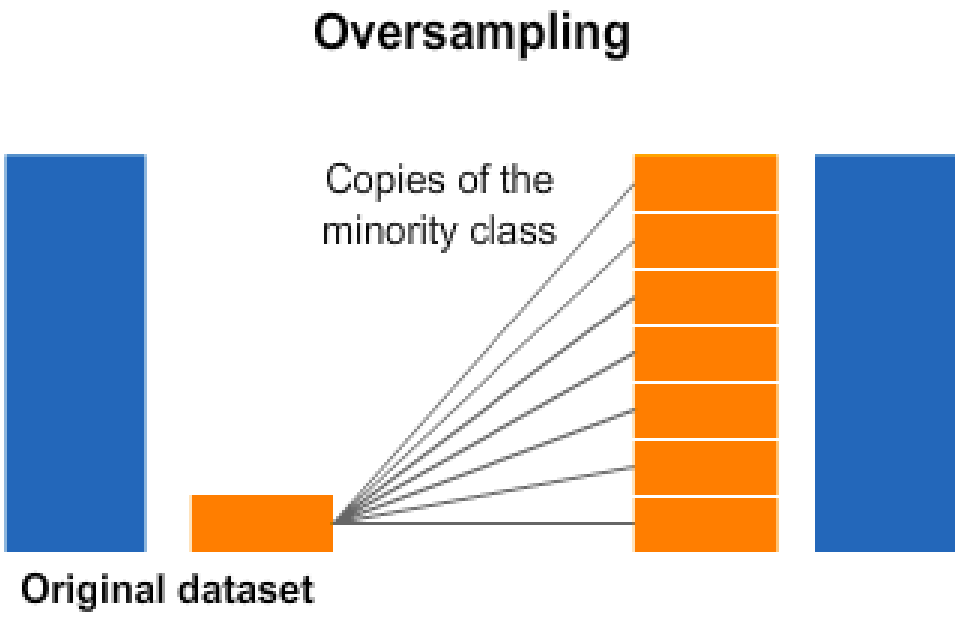

Рис. 19. Техника Oversampling для выравнивания классов

§3.3 Вид подаваемых на обучение данных

На Рис.20 можно видеть пример подаваемых на вход сети данных. Каждая строка описывает один кардиоцикл. В строчку последовательно записаны следующие величины:

- Длина интервала *QRS*;
- Длина интервала *QT*;
- Длина интервала *RR*;
- Длина интервала *PQ*;
- Амплитуда зубца *S*;
- Амплитуда зубца *P*;
- Амплитуда зубца *Q*;
- Направление зубца *T*;
- Размер зубца *Q*;
- Номер класса, которому принадлежит этот кардиоцикл (0 - больные инфарктом миокарда, 1 - больные аритмией, 2 - здоровые пациенты)

```
149 485 693  227    24  33 -124 -1 25 1
139 359 853  287  -125  1  -176  1 12 1
118 348 762  286  -208  8  -71  -1 14 1
108 533 1001 198  -16   16 -149 -1 10 2
127 543 1303 285 -452  -28 -205  1 11 2
163 724 1438 284 -465   -8 -202  1 12 2
```

Рис. 20. Вид подаваемых данных

§3.4 Обучение модели и полученные результаты

Для проверки результатов работы модели данные были поделены на обучающую и тестовую выборку в пропорции 80/20. Программный код построенной модели приведен в Приложении 2. Модель достигла точности распознавания 91% на обучающем и 90% на тестовом множестве. Для проверки результатов была построена матрица запутанности (confusion matrix) - матрица, позволяющая определить количество верно предсказанных результатов. Ее результаты приведены на Рис.21.

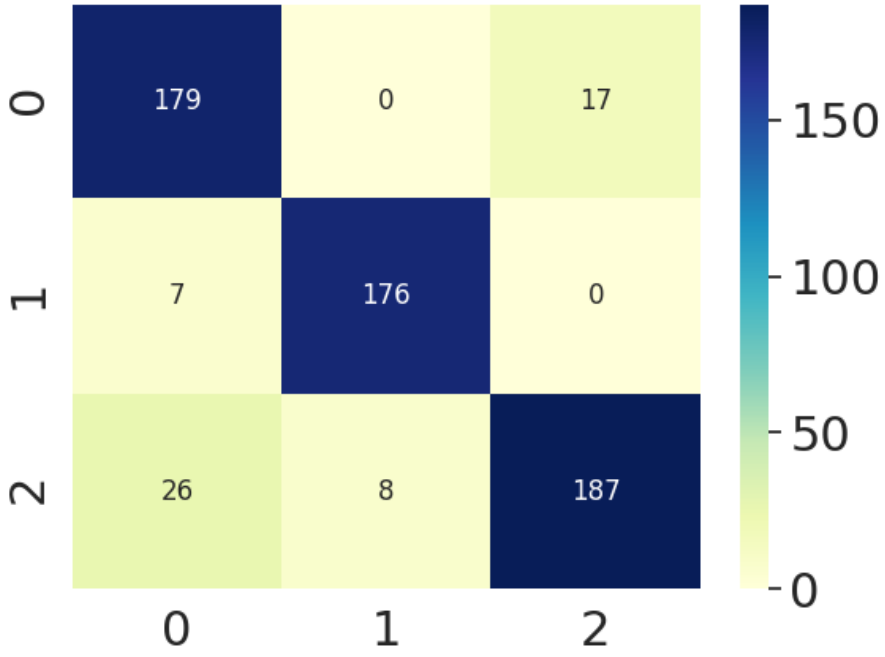

Рис. 21. Вид confusion matrix

По результатам, представленным в матрице, видно, что модель хуже всего различает классы 0 и 2 (инфаркт и здоровых людей), но в целом показывает приличные результаты.

На Рис.22 можно увидеть, как возрастала уверенность модели в прогнозе с продолжением обучения.

Рис. 22. Динамика точности модели в процессе обучения

Для подтверждения надежности модели была проведена кросс-валидация с использованием стратегии K-Fold (разделение данных на k равных частей). На каждой итерации модель показала результат, близкий к итоговому, что говорит о корректности построенной модели.

Заключение

В ходе данной работы была разработан программный комплекс для диагностики заболеваний по электрокардиограмме с использованием многослойного персептрона, одного из стандартных и хорошо зарекомендовавших себя механизмов решения задач классификации. Была решена задача препроцессинга искаженного сигнала, а также спроектирована модель нейронной сети для определения типа заболевания по ЭКГ.

Обученная система показывала точность более чем 90% на всех рассматриваемых классах - людях без заболеваний, с аритмией и с инфарктом миокарда.

Более совершенные алгоритмы препроцессинга кардиограмм позволят расширить круг детектируемых заболеваний и повысить точность распознавания, что говорит о том, что в данной области есть еще множество задач, которые необходимо решить. Узкая применимость разработанной модели в большой степени обусловлена достаточно ограниченным набором данных, поэтому любые добровольцы, готовые внести свой вклад в развитие этой области посредством пополнения датасета своими примерами кардиограмм, дали бы толчок к новым исследованиям. Также можно рассмотреть дополнительные отведения кардиограмм (II, III и грудные), что в перспективе позволит увеличить точность алгоритма.

Несмотря на достаточно высокую точность распознавания, всегда стоит стоит помнить о том, что система для распознавания заболеваний очень чувствительна к ошибкам, и любой неверный прогноз может стоить жизней многим пациентам. Но тем не менее, такие системы уже сейчас могут использоваться в современных медучреждениях для увеличения скорости

распознавания заболеваний, а значит - повышения эффективности системы здравоохранения.

Приложение

1. Программный код алгоритма детрендинга на языке программирования MATLAB

```
function [detrendedECG] = detrendECG(ECG)


[p,s,mu] = polyfit((1:numel(ECG))',ECG,10);
f_y = polyval(p,(1:numel(ECG))',[],mu);
detrendedECG = ECG - f_y;


end
```

2. Программный код реализованной нейронной сети на языке программирования Python

```
from keras.models import Sequential
from keras.layers import Dense, BatchNormalization, LeakyReLU
from keras.regularizers import l2
from sklearn.model_selection import train_test_split
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
import numpy
from imblearn.over_sampling import RandomOverSampler

numpy.random.seed(2)

def load_data():
    dataset = numpy.loadtxt("data.txt", delimiter=" ", usecols=range(10))
    X = dataset[:, 0:9]
    Y = dataset[:, 9]
    ros = RandomOverSampler()
    X, Y = ros.fit_sample(X, Y)
    return X, Y
```

```
def create_multi_label(Y):
    Y_new = []
    for i in range(0, len(Y)):
        if Y[i] == 0:
            Y_new.append([1, 0, 0])
        if Y[i] == 1:
            Y_new.append([0, 1, 0])
        if Y[i] == 2:
            Y_new.append([0, 0, 1])
    Y = numpy.array(Y_new)
    return Y

def create_model():
    model = Sequential()
    model.add(Dense(16, input_dim=9))
    model.add(BatchNormalization())
    model.add(LeakyReLU())
    model.add(Dense(32, activation='relu', activity_regularizer=l2(0.01)))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

def train_and_evaluate_model(model, X_train, Y_train, X_test, Y_test):
    history = model.fit(X_train, Y_train, epochs=200, batch_size=32,
    validation_data(X_test, Y_test), shuffle=True)
    score = model.evaluate(X_test, Y_test, batch_size=32)
    print("\n%s: %.2f%%" % (model.metrics_names[1], score[1] * 100)
    return history, (score[1] * 100)

def cross_validation(X, Y):
    kFold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    cross_validation_acc = []
    for train, test in kFold.split(X, Y):
        X_train= X[train]
        X_test = X[test]
        Y_train = create_multi_label(Y[train])
        Y_test = create_multi_label(Y[test])
        model = None
        model = create_model()
```

```
        history, acc = train_and_evaluate_model(model, X_train, Y_train, X_test, Y_test)
        cross_validation_acc.appendacc)
    for i in range(0, len(cross_validation_acc)):
        print('Model accuracy on split {} is {:.1f}%'.format(i + 1, cross_validation_acc[i]))


if __name__ == "__main__":
    X, Y = load_data()

    model = None
    model = create_model()
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
    Y_test_labels = Y_test

    Y_train = create_multi_label(Y_train)
    Y_test = create_multi_label(Y_test)

    history, acc = train_and_evaluate_model(model, X_train, Y_train, X_test, Y_test)

    classes = [0, 1, 2]
    y_pred = model.predict_classes(X_test)
    conf_matrix = confusion_matrix(Y_test_labels, y_pred)
    df_cm = pd.DataFrame(conf_matrix, range(3), range(3))
    sn.set(font_scale=2)  # for label size
    sn.heatmap(df_cm, annot=True, annot_kws={"size": 12}, fmt='d', cmap="YlGnBu")
    plt.show()

    plt.figure()
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('model loss')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['train', 'test'], loc='best')
    plt.show()

    plt.figure()
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
```

```
plt.title('model accuracy')
plt.ylabel('acc')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='best')
plt.show()
```

Список использованной литературы

1. М. Габриэль Хан - Быстрый анализ ЭКГ // (1999, Изд-во БИНОМ, Нев. диалект), 408 с.
2. Струтынский А.В. - Электрокардиограмма: анализ и интерпретация // Издательство Медпресс Россия, ISBN: 978-5-00030-383-2, 2016 г.
3. Д. Саймон - Карманный справочник по ЭКГ // Издательство: ГЭОТАР-Медиа Россия, ISBN: 978-5-9704-5453-4, 2020 г.
4. Rezaul Begg, Daniel T.H. Lai, Marimuthu Palaniswami - Computational Intelligence in Biomedical Engineering // ISBN 9780367388096, Published October 18, 2019 by CRC Press, 392 p.
5. Abdulhamit Subasi - Practical Guide for Biomedical Signals Analysis Using Machine Learning Techniques (A MATLAB Based Approach), Chapters 1-5 // 19th March 2019, 456 p.
6. Andrew Weems, Mike Harding, Anthony Choi - Classification of the ECG Signal Using Artificial Neural Network, Chapter 70 // International Conference on Electrical, Computer and Communication Technologies Conference, March 2017
7. Jia Li, Yujuan Si, Tao Xu, Saibiao Jiang - Deep Convolutional Neural Network Based ECG Classification System Using Information Fusion and One-Hot Encoding Techniques // Hindawi Mathematical Problems in Engineering Volume 2018, Article ID 7354081, 10 p.
8. Л. С. Фанзильберг - Компьютерная диагностика по фазовому портрету электрокардиограммы // Международный научно-учебный центр информационных технологий и систем НАН и МОН Украины, 190 с.
9. Воронцов Константин Вячеславович - Теория и практика машинного обучения: Задача диагностики заболеваний по электрокардиограмме // Летняя школа Яндекса, 2014

10. Мустафаев А.Г., Темирбулатов М.А., Омаров Р.С. - Определение аномалий сердечного ритма и выявление заболевания сердца при помощи нейронных сетей // Труды XVI Всероссийской конференци DICR-2017, Новосибирск, 4-7 декабря 2017 г.
11. Мурашко В.В., Струтынский А.В. - Электрокардиография // Издательство Медпресс Россия, ISBN: 978-5-00030-733-5, 2020 г.
12. Контурный анализ ЭКГ [Электронный ресурс], URL: http://www.biors.ru/tech/practicing-biors/konturniy-analiz-ekg.htm
13. База данных PTB Diagnostic ECG Database [Электронный ресурс], URL: https://www.physionet.org/content/ptbdb/1.0.0/
14. База данных MIT-BIH Arrythmia Database [Электронный ресурс], URL: https://www.physionet.org/physiobank/database/mitdb
15. Resampling Strategies in Imbalanced Datasets [Электронный ресурс], URL: https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets
16. Selcan Kaplan Berkaya, Alper Kursat Uysal, Efnan Sora Gunal, Semih Ergin, Serkan Gunal, M. Bilginer Gulmezoglu - A survey on ECG analysis, Biomedical Signal Processing and Control // Volume 43, 2018, Pages 216-235, ISSN 1746-8094
17. Udit Satija, Barathram Ramkumar, M. Sabarimalai Manikandan - An automated ECG signal quality assessment method for unsupervised diagnostic systems, Biocybernetics and Biomedical Engineering // Volume 38, Issue 1, 2018, Pages 54-70, ISSN 0208-5216
18. Aykut Diker, Derya Avci, Engin Avci, Mehmet Gedikpinar - A new technique for ECG signal classification genetic algorithm Wavelet Kernel extreme learning machine // Optik, Volume 180, 2019, Pages 46-55, ISSN 0030-4026

19. 10 ведущих причин смерти в мире [Электронный ресурс], URL: https://www.who.int/ru/news-room/fact-sheets/detail/the-top-10-causes-of-death
20. В.В.Вьюгин - Математические основы теории машинного обучения и прогнозирования // Москва, 2013 г., 305 с.
21. Осовский С. - Нейронные сети для обработки информации // Пер. с польского И.Д. Рудинского. - М.: Финансы и статистика, 2002. - 344 с.